Advanced Analytics mit der analytischen In-Memory Datenbank EXASolution

Sebastian Klenk, Stefan Mandl, Simon Zentgraf, Mathias Golombek

EXASOL AG
Neumeyerstraße 48
90411 Nürnberg
sebastian.klenk@exasol.com
stefan.mandl@exasol.com
simon.zentgraf@exasol.com
mathias.golombek@exasol.com

Abstract: Aufgrund der kontinuierlich steigenden Datenmengen müssen immer aufwendigere Berechnungen in der Datenbank durchgeführt werden. Dies wird umso wichtiger je größer die Datenmengen werden und je spezieller die Berechnungen sind. Gerade im Umfeld von Big Data und In-Memory sind komplexe statistische Berechnungen auf Datenvolumina, die gängige Hauptspeicherumfänge bei weitem übersteigen, an der Tagesordnung. EXASOL hat mit seiner Advanced-Analytics-Umgebung EXAPowerlytics ein Werkzeug geschaffen, das es erlaubt, komplexe Berechnungen in der Datenbank durchzuführen. Dabei vereint EXAPowerlytics die Vorteile einer modernen In-Memory Datenbank mit denen gängiger Programmierumgebungen: (1) Es ist vollständig in SQL integriert und damit standardkonform, (2) es erlaubt die Entwicklung in den unterschiedlichsten Programmiersprachen und (3) es erlaubt die schnelle und einfache Entwicklung massiv paralleler Algorithmen.

1 In-Memory und Big Data

Bis zum Ende 2013 wird die Menschheit mehrere Trilliarden (10^{21}) Bytes an Daten und Informationen ansammeln [MC13] – eine nahezu unvorstellbar große Datenmenge. Kein Wunder in Anbetracht des enormen Wachstums des Internet und des stetig steigenden Datenuploads in soziale Medien. Aber auch in der Industrie schießt das Datenaufkommen vor allem durch Maschinen-zu-Maschinen-Kommunikation (M2M-Kommunikation) und langfristig gespeicherte Prozess- und Messdaten in die Höhe. Datensammlungen, die vor wenigen Jahren noch ein Segen für Unternehmen waren, wandelten sich innerhalb der letzten Jahre zu einer großen Herausforderung. Dieser Wandel spiegelt sich in diversen Faktoren wider, die ein Umdenken im Umgang mit großen Datenmengen fordern: Zum einen können bislang eingesetzte Entscheidungsverfahren nicht mehr angewendet werden, da bisher genutzte Algorithmen nur bei kleinen Datenmengen effektiv einsetzbar sind. Der extreme Datenzuwachs

bremst heutzutage die veralteten Algorithmen aus – kurzfristige Datenanalysen sind so unmöglich. Zum anderen wächst mit dem Datenumfang auch der Anspruch an die durchgeführten Analysen. So reichte es beispielsweise seither aus, bei Umsatzanalysen den Umsatz eines Monats oder eines Jahres für Geschäftsberichte auswerten zu lassen. Heutzutage müssen besonders in der produzierenden Industrie aktuelle Datensätze in Echtzeit analysiert und ausgewertet werden. Es reicht nicht mehr aus, dass am Ende eines Tages eine ausführliche Auswertung des Vortags vorliegt. Produktions-Reports und Qualitätskontrollen müssen ad hoc abrufbar sein. Ein dritter Faktor, der den durch wachsende Datenmengen hervorgerufenen Wandel beeinflusst, ist die fehlende Nutzbarkeit bisheriger Systeme. Vor allem unstrukturierte Daten – etwa von Sensormessungen in der Produktion oder Social-Media-Daten – können nicht wie gewohnt analysiert werden. Heutzutage werden Daten wie Bilder und Texte zu Analysen herangezogen, die zunächst aufwändig in eine für die Analysedatenbank verständliche Datenform übersetzt werden müssen. Werden diese Daten mit strukturierten Daten verknüpft, erhält man eine umfangreiche Datenbank, die nicht selten aus mehreren Terabyte an Informationen besteht. Erst neue Techniken, zum Beispiel In-Memory und massiv paralleles Arbeiten, ermöglichen es, vorliegende Datenberge in kürzester Zeit auszuwerten.

In-Memory Datenbanken spiegeln eine Entwicklung wider, die Datenbanken zurück in das Zentrum strategischer Anwendungen stellt. Durch die immer größer werdenden Datenmengen ist es schlicht und einfach nicht mehr möglich große Teile eines Datensatzes für aufwändige Berechnungen aus der Datenbank in ein Anwendungssystem zu laden. Eine Folge daraus ist das In-Memory-Prinzip: Die Datenbank hält die Daten im Hauptspeicher und kann sie dadurch schneller ausliefern. Gerade für einfachere Berechnungen liefert diese Herangehensweise sehr viele Vorteile, da die Daten der Anwendung sehr schnell zur Verfügung stehen und nur wenige Änderungen an bestehenden Programmen notwendig sind. Für aufwendigere Berechnungen ist die In-Memory-Anbindung jedoch nicht schnell genug, da die Daten immer noch über die relativ langsame Anbindung der Datenbank an das Anwendungssystem übertragen werden müssen. Gerade aufwändige Statistiken oder komplexe Geschäftsanwendungen erfordern eine sog. Datenlokalität. Das heißt, die Berechnungen müssen dort stattfinden, wo die Daten abgelegt sind. Auf diesem Prinzip basiert auch EXAPowerlytics.

Die Integration von Datenverarbeitungsmethoden in die Datenhaltung ist schon seit Jahren ein zentrales Forschungsthema. Im Bereich analytischer Methoden und Data Mining Verfahren besteht schon lange der Wunsch eine effiziente Integration dieser beiden Welten zu entwickeln [Cha98]. Doch gerade für große Datenmengen scheint eine Verbindung über Systemgrenzen hinweg nicht der richtige Weg [Sar00]. An dieser Einsicht hat sich, unserer Meinung nach, bis heute nichts geändert. Daher konzentrieren wir uns bei unserer Arbeit darauf so viele Berechnungen wie möglich innerhalb, oder zumindest nahe, der Datenbank durchzuführen.

2 Berechnungen in der Datenbank

EXAPowerlytics erlaubt es, komplexe Algorithmen in der Datenbank auszuführen und dabei die Vorteile einer In-Memory-Datenbank mit denen moderner Programmiersprachen zu verknüpfen. Dies erfolgt durch die Einbindung der Programmiersprachen Lua [Lua96], Python [Py13] und R [R13]. Der Programmcode der jeweiligen Sprache kann direkt in SQL definiert werden.

```
DROP SCRIPT greet_world;
CREATE R SCALAR SCRIPT greet_world (name VARCHAR(50))
EMITS (greeting VARCHAR(100) AS

      run <- function(ctx) {
            ctx$emit(paste("hello world from ",name))
      }
/
```

Und von dort auch aufgerufen werden.

```
SELECT greet_world ('EXASOL') FROM DUAL;
```

Diese Art von Skript bezeichnen wir als User Defined Function (UDF). Durch die Ausführung von Teilen der Anwendungslogik in der Datenbank erhält man sehr schnellen Zugriff auf die Daten, was selbst die Verarbeitung von Datenmengen aus dem Big Data Umfeld möglich macht.

Es gibt vier verschiedene Arten von UDFs, die in EXAPowerlytics definiert werden können:

1. SCALAR-RETURNS: Eine Funktion, die ein Tupel skalarer Werte, also Zahlenwert, übergeben bekommt und einen einzelnen Zahlenwert zurückgibt.

2. SCALAR-EMITS: Eine Funktion, die ein Tupel skalarer Werte übergeben bekommt und eine Liste skalarer Werte zurückgibt.

3. SET-RETURNS: Eine Funktion, die eine Menge von Werten übergeben bekommt und einen einzelnen Wert zurückliefert.

4. SET-EMITS: Eine Funktion, die eine Menge von Werten übergeben bekommt und eine Menge von Werten zurückliefert.

Jede dieser vier Arten hat ihre Anwendungsfälle. SCALAR-RETURNS wäre zum Beispiel für die Berechnung der Addition zweier Werte geeignet. Mit SCALAR-EMITS ließe sich beispielsweise die Folge der Fibonnacci-Zahlen bis zu dem übergebenen Wert bestimmen. SET-RETURNS eignet sich für MIN/MAX oder MEDIAN Operationen, bei denen aus einer Liste von Werten ein Element bestimmt oder berechnet werden soll. SET-EMITS ist für Filteroperationen oder für statistische Operationen geeignet, bei

denen zu jedem Element einer Menge eine Wahrscheinlichkeit in Abhängigkeit der Gesamtmenge zugewiesen werden soll.

2.1 Arbeiten wie am Fließband

Die Bearbeitung von SQL-Abfragen findet in EXASolution in Form sich überlappender Arbeitsschritte statt. Diese Vorgehensweise ist bekannt als Pipelining [GUW08] da, ähnlich wie die Teile auf einem Fließband die Daten durch eine Folge von Stationen geschoben werden in denen sie verarbeitet werden. Eine dieser Stationen ist zum Beispiel ein Filter, gegeben durch das WHERE-Statement einer SQL Abfrage, der nur die Datenelement durchlässt, die zur Abfrage passen.

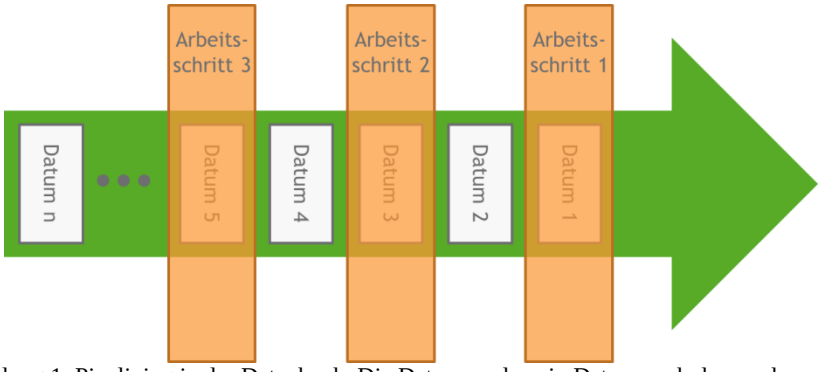

Abbildung 1: Pipelining in der Datenbank. Die Daten werden ein Datum nach dem anderen durch den Verarbeitungsprozess geschoben. Dabei werden die Arbeitsschritte auf die einzelnen Datensätze angewendet was ein temporäres Abspeichern des gesamten Datensatzes erübrigt.

Eine Station in der Fließbandverarbeitung ist auch die Ausführung einer UDF. Das bedeutet, dass eine Funktion, sofern es sinnvoll ist, immer nur einen kleinen Teil des gesamten Datensatzes betrachten muss. Gerade bei sehr großen Datenmengen hat dies den Vorteil, dass nicht der gesamte Datensatz in den Hauptspeicher geladen werden muss um die Funktion auszuführen, sondern nur der aktuelle Datensatz. Die Funktion selbst verarbeitet die Daten dann ebenfalls in Fließbandmanier: In einer Schleife wird immer das jeweils aktuelle Datum gelesen, verarbeitet und ausgegeben. Dieser Vorgang wiederholt sich so lange, bis alle Datenelemente verarbeitet sind.

```
CREATE LUA SET SCRIPT pipeline (x DOUBLE)
      EMITS (x DOUBLE) AS
            function run(data)
                  local current_result = 0
                  Repeat
```

```
                -- perform operation on current data item
                   current_result = do_something(data.x)

                -- write result of operation back 2 pipeline
                   data.emit(current_result)

                   -- fetch next data item and iterate
             until not data.next()
             end
/
```

Somit kann eine UDF mehrere hunderte Millionen von Datensätzen verarbeiten ohne den gesamte Datensatz geladen haben zu müssen. Gerade im Big Data Umfeld ein unschätzbarer Wert.

3 Massiv parallele Berechnungen in der Datenbank

Die Ausführung von Operationen auf einem Datensatz ist nur ein kleiner Teil der Mächtigkeit von EXAPowerlytics. Die besondere Leistungsfähigkeit der Umgebung zeigt sich in der parallelen Ausführung von Skripten auf mehreren Knoten.

3.1 Aufbau von EXASolution

EXASolution ist als Cluster-System konzipiert. Das heißt, das Datenbanksystem verteilt die Daten auf alle Knoten im Cluster, sodass jeder einzelnen Rechner nur einen Teil der Daten zu verwalten hat. Wenn eine SQL-Abfrage an das System gestellt wird, greift das System auf alle Knoten zu und verteilt auf diese Weise die Last der Abfrage auf das gesamte Cluster. Je besser die Verteilung der Daten dem Charakter der Abfragen entspricht umso ausgeglichener ist die Lastverteilung.

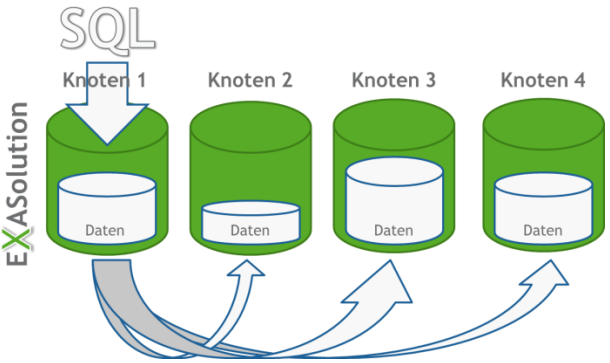

Abbildung 2: Parallele Ausführung einer Abfrage in EXASolution. Die Datenbank verteilt die Abfrage auf alle Knoten und jeder Knoten bearbeitet die Abfrage unabhängig von den anderen Knoten.

Dieses Prinzip der Verteilung der Last auf die einzelnen Rechnerknoten ist nicht nur für die Beantwortung von Abfragen sinnvoll, es ist besonders gut für die Durchführung komplexer Berechnungen geeignet. So lässt sich zum Beispiel bei einer einfachen Mittelwertberechnung, in einem ersten Schritt, der Mittelwert über die einzelnen Teilmengen unabhängig voneinander und parallel berechnen und, in einem zweiten Schritt, der Mittelwert über die (gewichteten) Mittelwerte bestimmen. Diese Vorgehensweise, (1) das Verteilen der Berechnung auf eine Reihe von Computern und (2) das Zusammenführen der einzelnen Ergebnisse zu einem Gesamtergebnis ist bekannt unter dem Namen MapReduce (MR) [DG04]. MR ist vor allem wegen seiner Einfachheit populär da es im Gegensatz zu anderen Konzepten zur verteilten Ausführung von Berechnungen, die meisten kommen aus dem wissenschaftlichen Rechnen [KK02], ist MR bestechend einfach und trotzdem sehr vielseitig einsetzbar.

Durch EXAPowerlytics ist die Entwicklung verteilter Algorithmen noch einmal leichter geworden. Die Datenbank übernimmt dabei die Verteilungsarbeit und der Entwickler muss sich nur noch die einzelnen Berechnungen ausgestalten.

```
CREATE LUA SET SCRIPT group_max (x DOUBLE,id VARCHAR(10))
      EMITS (x DOUBLE, id VARCHAR(10)) AS
            function run(data)
                  local max = 0
                  local id = ''
                  repeat
                        if data.x >= max then
                              max = data.x
                              id = data.id
                        end
                  until not data.next()
                  data.emit(max,id)
            end
/

SELECT group_max(x,id) FROM (SELECT avg(x) AS x,id FROM tt
GROUP BY id);
```

Dieses Beispiel zeigt die parallele Berechnung des Mittelwerts mit Hilfe der eingebauten Funktion AVG im Map-Schritt und die Aggregation der Werte mit Hilfe des Lua-Skripts GROUP_MAX im Reduce-Schritt. Die Datenbank verteilt dabei die Gruppen, die durch das GROUP BY-Statement entstehen, auf die einzelnen Knoten und die Berechnung des Mittelwerts wird dann parallel auf den jeweiligen Knoten unabhängig von den Mittelwerten der anderen Gruppen bestimmt. Erst die Berechnung des Maximums von allen Gruppen-Mittelwerte findet sequenziell auf einem Knoten statt. Diese Form der parallelen Ausführung bringt gerade bei komplexen Map-Schritten und einfachen Reduce-Schritten erhebliche Geschwindigkeitsvorteile.

Ein Beispiel für eine solche Anwendung wäre zum Beispiel die Auswahl von Derivaten auf Basis zugrundeliegender Aktienkurse. Die Tabellenwerte wären die historischen Werte der verschiedenen Aktien. Jede Aktie entspricht dann einer Gruppe über die aggregiert wird. Als Map-Funktion nutzt man einen Preisfindungsalgorithmus (hierfür gibt es in R zahlreiche Pakete) und in der Reduce-Funktion sucht man sich die Derivate mit den im Augenblick günstigsten Eigenschaften aus. Da es sehr viele Aktien gibt und die Berechnung eines fairen Preises sehr rechenaufwändig ist, erhält man durch die Parallelisierung einen Geschwindigkeitsgewinn, der linear mit der Anzahl der beteiligten Computer steigt.

4 Externe Programme einbinden

Die Auswahl an Programmiersprachen, die für die Entwicklung von UDFs eingesetzt werden kann, ist oft maßgeblich entscheidend für die Akzeptanz einer Datenbanklösung. EXASolution erlaubt es, Programme, die in beliebigen Programmiersprachen entwickelt wurden, als UDFs einzubinden. Die Programme laufen dabei als eigenständige Prozesse und die Kommunikation mit der Datenbank findet über ØMQ [H13] statt. Da ØMQ auch

den Zugriff auf nicht-lokale Prozesse ermöglich, also auf Prozesse, die auf einem System das nur über das Netzwerk mit den Knoten der Datenbank verbunden ist, sind der Fantasie kaum Grenzen gesetzt. Von lokal ablaufenden Java-Programmen bis hin zum Zugriff auf Legacy-Software, die auf Host-Systemen läuft, ist fast alles möglich. Durch diese Herangehensweise lassen sich unterschiedliche System sehr leicht in ein einheitliches Abfragesystem integrieren.

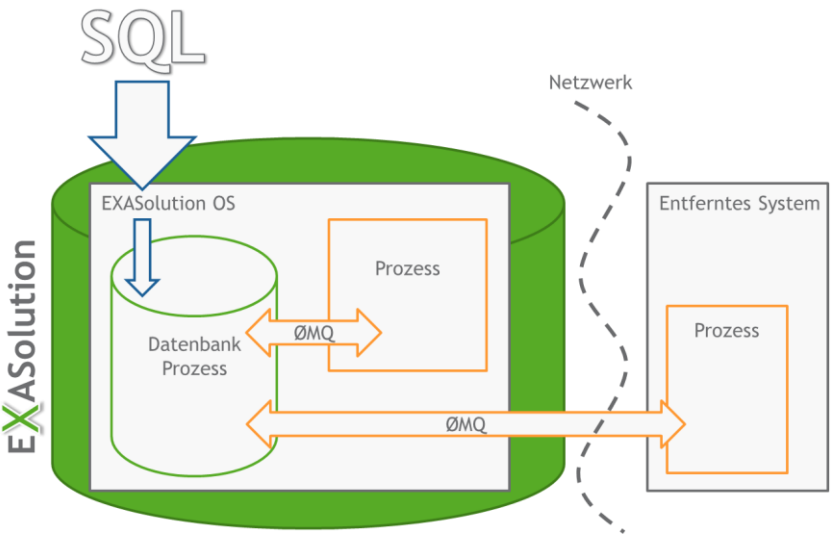

Abbildung 3: Die Anbindung externer Prozesse an EXASolution. EXASolution ermöglicht die Anbindung durch 0MQ. Dabei ist es unerheblich, ob der Prozess auf dem lokalen Knoten läuft oder auf einen entfernten System.

Um eine externe UDF (eUDF) zu definieren, reicht es daher aus, die Signatur der Funktion, also deren Namen und deren Übergabe- und Rückgabeparameter, und die Netzwerkadresse (IP-Adresse und TCP-Port) unter der der Dienst zu finden ist, anzugeben.

```
CREATE EXTERNAL SCALAR SCRIPT external_script(param DOUBLE)
EMITS (num INT) AS
# redirector tcp://192.168.1.1:4050
/
```

Dieser kurze Abschnitt gibt an, dass unter der Adresse 192.168.1.1:4050 eine externe UDF zu finden ist. Das korrespondierende Programm auf der Seite des externen Prozesses implementiert dann einen Dienst der an der angegebenen Adresse auf

Anfragen wartet und diese dann ausführt. Die Implementierung eines solchen Dienstes ist zwar aufwändig aber auch sehr flexibel. Mit Hilfe dieser Herangehensweise können nicht nur UDFs in beinahe beliebigen Programmiersprachen entwickelt werden, es ist auch ein Leichtes Dienste wie zum Beispiel Web-Services oder bestehende Programme anzubinden. In einem solchen Fall würde dann die eUDF nur als Hülle für den eigentlichen Dienst dienen.

4.1 Hadoop und die Welt der unstrukturierten Daten

Eine besondere Art externer Prozesse ist sicher der Hadoop Map-Reduce Job. Er erlaubt die einfache Verarbeitung unstrukturierter Daten mithilfe des gleichnamigen Open Source Frameworks und hat daher in den letzten Jahren enorm an Popularität gewonnen. Mit Hilfe des EXAPowerlytics Hadoop Integration Service lassen sich Hadoop Jobs einfach in die SQL-Verarbeitung von EXASolution einbinden. Dieser Service erweitert das eUDF Framework um einen Verwaltungsdienst, den Redirector Service. Dieser verwaltet die Zuordnung von Hadoop- zu EXASolution-Knoten und umgekehrt. Abbildung 4 zeigt den Aufbau des Hadoop Integration Services. Mit Hilfe des SQL Befehls

```
CREATE EXTERNAL SCALAR SCRIPT external_script(param DOUBLE)
EMITS (num INT) AS
# redirector tcp://192.168.1.1:4050
exasol.method = input
exasol.input.splits = 1
mapred.output.dir = file:/resultdir
mapreduce.job.jar = /nfs/home/my_user/my_exa_classes.jar
mapreduce.job.name = hd_job
mapreduce.job.map.class = EXAHdJobMapper
mapreduce.job.inputformat.class =
exasol.hadoop.EXAIOFormat$Input
mapreduce.job.output.key.class =
org.apache.hadoop.io.IntWritable
mapreduce.job.output.value.class =
org.apache.hadoop.io.Text
mapreduce.job.output.fileoutputformat.outputdir =
file:/resultdir
mapreduce.job.outputformat.class = \
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
mapreduce.job.maxtaskfailures.per.tracker = 0
mapreduce.job.maps = 16
mapreduce.map.maxattempts = 1
/
```

wird eine Referenz auf einen existierenden Hadoop Job angelegt der dann direkt aus der SQL-Verarbeitung heraus aufgerufen werden kann. Neben der Definition des Hadoop Jobs in EXASolution muss auch noch der eigentliche Hadoop Job programmiert werden. Dabei unterscheidet sich ein Hadoop Job für EXASolution nur in wenigen Aspekten von

einem regulären Hadoop Job. Der wesentliche Unterschied besteht in der Art wie die Daten zwischen den Java Klassen und der Datenbank ausgetauscht werden. Hierfür stellt der EXAPowerlytics Hadoop Integration Service eine Reihe von Hilfsklassen zur Verfügung. Das nachfolgende Quelltextabschnitt zeigt ein Mapper-Skelet für die Integration der beiden Dienste.

```
public class EXAHdJobMapper {

    public static class InputMapper
          extends Mapper<EXAIOFormat.Key,
                         EXAIOFormat.Value,
                         LongWritable, Text> {
        /* ... */

        public void map(EXAIOFormat.Key key,
                        EXAIOFormat.Value value,
                        Context context)
                 throws IOException, InterruptedException {

           Double input;

           for (boolean rowsLeft=true; rowsLeft;
                rowsLeft=value.readNextRow(false)) {
                input = value.getDouble(0);
                value.newResultRow()
                     .setInteger(Math.round(input)).emit();
           }
        }
     }
}/
```

Durch die Zusammenarbeit von EXASolution und Hadoop ist es möglich unstrukturierte und strukturierte Daten gemeinsam zu verarbeiten. Auf diese Weise können die unterschiedlichsten Datenquellen mit Unternehmensdaten verknüpft werden. Zum Beispiel können Daten aus sozialen Medien oder Click-Stream-Daten mit Kundendaten verknüpft werden, Kundenbewegungsdaten können mit Verkaufszahlen verknüpft oder Bilddaten können mit Daten der Qualitätssicherung verknüpft werden. Erst durch die Verknüpfung der unterschiedlichen Datenquellen entsteht ein tiefergehender Mehrwert.

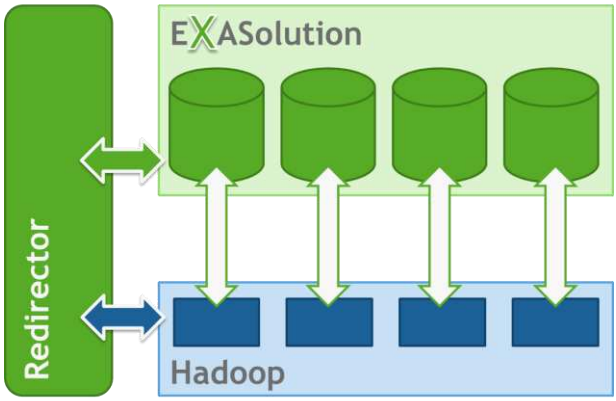

Abbildung 4: Die Integration von EXASolution und Hadoop. EXASolution greift über den Redirector Dienst auf Hadoop zu. Der Redirector übernimmt dabei nur die Vermittlung, die eigentliche Kommunikation und Datenübertragung erfolgt direkt zwischen den Knoten von EXASolution und Hadoop.

5 Zusammenfassung

Komplexe analytische Berechnungen werden in immer größerem Umfang in der Datenbank durchgeführt. Dies liegt zum einen daran, dass die Mengen an Daten Größenordnungen angenommen haben die den Hauptspeicher selbst größter Rechner übersteigen und zum anderen an den wachsenden Analysemöglichkeiten moderner Datenbanksysteme. Die Fähigkeiten der analytischen Datenbank EXASolution und des Advanced Analytics Frameworks EXAPowerlytics bieten in diesem Zusammenhang ein umfangreiches Spektrum an Optionen komplexe Berechnungen in der Datenbank durchzuführen. Dazu gehört die Ausführung von Skripten, geschrieben in den Sprachen Lua, R oder Python, die massiv-parallele Verteilung dieser Skripte und die Anbindung externer Prozesse wie Web-Services oder Host-Systeme. Darüber hinaus bietet die Integration von Hadoop die Möglichkeit strukturierte und unstrukturierte Daten gemeinsam zu analysieren.

EXASolution ist ein modernes analytisches In-Memory Datenbankmanagementsystem, dass es erlaubt komplexe Berechnungen in der Datenbank durchzuführen. Dies ermöglicht es businessrelevante Analysen auf großen Datenmengen, jenseits hunderter Terabyte, durchzuführen und somit Fragestellungen zu beantworten die vorher nicht beantwortet werden konnten.

Literaturverzeichnis

[MC13] Mayer-Schönberger, V.; Cukler, K: Big Data. John Murray, London, 2013.

[Cha98] Chaudhuri, S: Data Mining and Database Systems: Where is the intersection? IEEE Data Engineering Bulletin 1989;

[Sar00] Sarawagi, S; Thomas, S; Agrawal, R. Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications: Data Mining and Knowledge Discovery 2000 (4), S 89-125

[GUW08] Garcia-Molina, H; Ullman, J.; Widom J.: Database Systems: The Complete Book (2 ed.). Prentice Hall Press, Upper Saddle River, NJ, USA, 2008.

[Lua96] Ierusalimschy, R.; de Figueiredo, L. H.; Celes, W.: Lua - an extensible extension language. Software: Practice & Experience 26 1996; S. 635–652.

[R13] R Core Team: R: A Language and Environment for Statistical Computing, Vienna, Austria, 2013. http://www.R-project.org

[Py13] Python Software Foundation: Python Programming Language, Beaverton, OR, USA, 2013. http://www.python.org

[DG04] Dean, J.; Ghemawat S.: MapReduce: Simplified Data Processing on Large Clusters OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

[KK02] Karniadakis, G.; Kirby, R.: Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation. Cambridge University Press, 2002.

[H13] Hintjens, P.: ZeroMQ - Messaging for Many Applications. O'Reilly Media, Sebastopol, CA, USA, 2013